Class name : SY CSE(IOT)
Rollno : 2007
Urn no : 1022101007
Batch : S1

EXPERIMENT 7
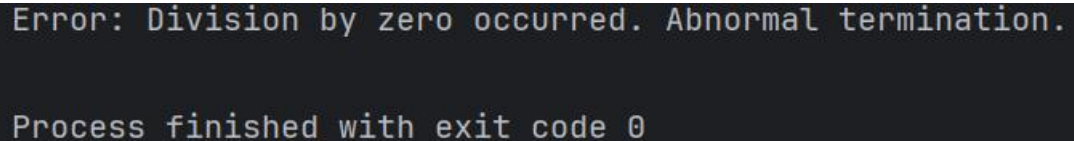
7. Handle Exceptions using python built in Exceptions

a) Demonstrate a python code to implement abnormal termination?

CODE:-
```python
def divide_by_zero():
    result = 10 / 0
try:
    divide_by_zero()
except ZeroDivisionError:
    print("Error: Division by zero occurred. Abnormal termination.")
```

OUTPUT:-

b) Demonstrate a python code to print try, except and finally block statements.

CODE:-
```python
def divide(x, y):
    try:
        result = x / y
        print(f"Result of division: {result}")
    except ZeroDivisionError:
        print("Error: Division by zero occurred.")
    finally:
        print("Finally block executed.")

print("Test Case 1:")
divide(10, 2)

print("\nTest Case 2:")
```

```
divide(10, 0)

print("\nTest Case 3:")
divide("10", 2)
```

OUTPUT:-

```
Test Case 1:
Result of division: 5.0
Finally block executed.

Test Case 2:
Error: Division by zero occurred.
Finally block executed.

Test Case 3:
Finally block executed.
Traceback (most recent call last):
  File "D:\Python\Experiments\Exp 7\Exp7b.py", line 17, in <module>
    divide("10", 2)
  File "D:\Python\Experiments\Exp 7\Exp7b.py", line 3, in divide
    result = x / y
TypeError: unsupported operand type(s) for /: 'str' and 'int'

Process finished with exit code 1
```

c) Write a program to create user defined exception in python.

CODE:-
```python
class CustomError(Exception):
    def __init__(self, message):
        self.message = message
def validate_age(age):
    if age < 18:
        raise CustomError("Age must be 18 or above.")
try:
    user_age = int(input("Enter your age: "))
    validate_age(user_age)
    print("Age is valid.")
except CustomError as e:
    print(f"Custom Error: {e.message}")
except ValueError:
    print("Error: Please enter a valid integer for age.")
```

OUTPUT:-

```
C:\Users\Dragon\AppData\Local\Programs\
Enter your age: 10
Custom Error: Age must be 18 or above.


Process finished with exit code 0
```